We normally publish the post-mortem for an assignment after it has been marked and released. Here is a list of common errors provided by the graders for assignment 4.

**General**

- Many students were missing examples for base case(s) throughout the assignment. When writing a recursive function, there should be an example for every base case, and an example for at least one of the recursive cases, as discussed in the style guide.

- Contracts that consumed or produced lists often had formatting errors. For example, if a function consumes a list of numbers, it should be indicated as `(listof Num)` in the contract **exactly**, and not `listof(Num)`, `list of Num`, `list-num`, `list`, or any other variation.

**Question 2**

- Many students did not define constants for the `Rental` rates or `Rental` capacities associated with a `Bicycle`, `Boat`, or `Horse`.

- Many students did not include a contract for their `rental-template`. As discussed in the style guide, templates should always include a contract.

- Many students had templates that differed greatly from the solutions. The `rental-template` should look similar to the `mminfo-template` on slide 32 of module 4.

- In part (c), some students did not make sure that a `Horse` rental was valid only if the consumed number of renters did not exceed the capacity of a `Horse`, and that the consumed duration did not exceed the `Horse`’s stamina.

- In part (d), many students indicated that the function produces a `Num`, but the most specific type in this case would have been a `Nat`.

- In part (d), some students did not correctly communicate in their requirements that the `Rental` must be valid according to the consumed parameters.

- In part (d), some students did not correctly account for the case where the number of renters is greater than 3 for a `Horse`.

**Question 3**

- In part (a), many students produced `true` if the largest even number in the consumed list was equal to the consumed number, when `largest-even?` should have produced `false`. 
• In part (a), some students did not include the requirement that the consumed candidate integer must be even.

• In part (c), some students kept any 0's that were in the consumed list, when the specifications stated that only positive numbers should be represented in the produced list.

• In part (c), some students did not include the requirement that the consumed height must be positive.

• In part (d), many students produced true if the consumed list was (cons 0 empty), when the function should have produced false upon seeing a zero.

Question 4

• Many students did not define the house radius of 183 as a constant.

• Some students did not use count-less-than as a helper function in part (b), which often lead to incorrect or overly complex solutions.

• In part (b), many students were missing at least one of the requirements in the context of this question.

• In part (b), some students did not correctly account for the fact that some of the elements in the consumed lists could be greater than or equal to 183, which would not count in the final score.

Ongoing Errors

The following is a list of common errors from previous assignments that were still repeated for assignment 4.

• Many students are still missing parameter references in their purpose statements. Purpose statements should meaningfully use each parameter name, and the parameter names should be written exactly as they appear in the function header.

• Constants (and helper functions) should be defined above the design recipe for the function they are used in. Some students still defining their constants and helper functions between their examples and function definition, or after the main functions specified in the assignment.

• All design recipe components, except for tests, are required for helper functions. Some students did not include these design recipe components for their helper functions. item For now, equal? should only be used to compare two values of unknown types, or values that can take on more than one type. When the types of the arguments are known, use the most appropriate comparison function (e.g., symbol=?).

• Starting a new cond expression immediately in an else clause is unneeded. Instead, directly check for the next condition in the original cond expression.